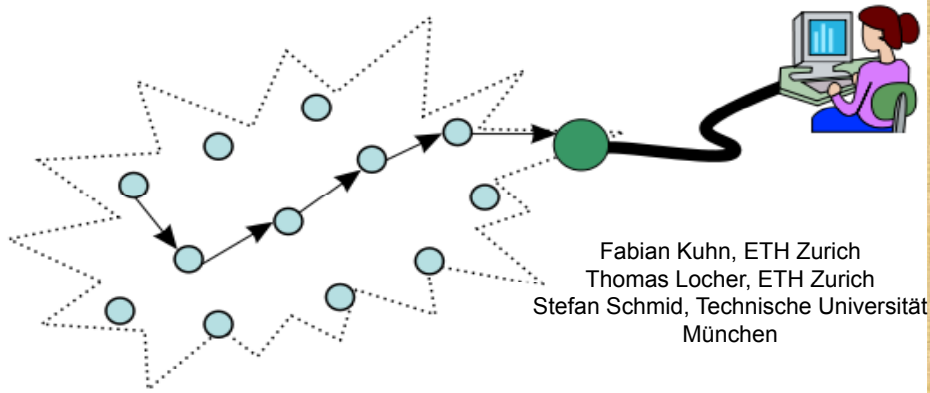


# Distributed Computation of The Mode



Distributed Algorithms,  
0510-7406

Prof. Boaz Patt-Shamir  
Tel-Aviv Uni.

Presented By:  
B.Sc. Lior Zatlavi

1

Distributed Computation of The Mode

10 May 2010

## Scope

- Introduction - Problem & Motivation
- Related Work
- Model
- Solution(s)
  - Deterministic Algorithm
  - Randomized Algorithm
- Further Studying of Freq. Dist. Consideration
- Conclusion

2

Distributed Computation of The Mode

10 May 2010

## Introduction

So what are we talking about?

3

Distributed Computation of The Mode

10 May 2010

## Problem – General Description

- Input:
  - A graph / network of nodes holding elements
- Output:
  - The most frequent element in the graph (“Mode”)

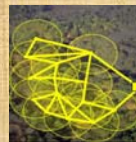
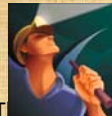
4

Distributed Computation of The Mode

10 May 2010

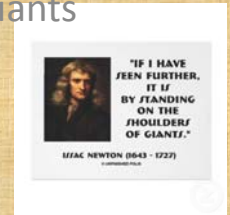
# Motivation

- Why is this important?
- Statistics about **non-centralized systems**
  - ISP / Cellular provider mapping its most frequent users
  - File sharing in **p2p** networks
  - **Knowledge about customers** for large franchises
  - Acquiring data from **wireless sensor networks**
  - Can be applied to **data mining**



# Related Work

Standing on the shoulders of giants



# Related Problems

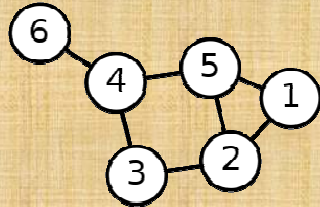
- Kuhn et al. – randomized algorithm for the  $k$ -selection problem –  $O(D \log_D n)$
- Flajolet et al. – LOGLOG algorithm for finding the number of distinct elements –  $O(D)$
- Alon et al. – estimation of second moment  $F_2$  –  $O(D)$

# Mode Computation Problem

- Munro et al., 1980 – an algorithm which needs  $n \log(n/m) + o(n \log(n/m)) + O(n)$  comparisons
  - $n$  – total number of elements
  - $m$  – frequency of the mode
- Farzan et al., 2005 – Cache-Oblivious Comparison-Based Algorithm on Multisets
- Charikar, Chen, and Farach-Colton, 2004 – finding  $k$  elements with  $(1 - \epsilon)m_k$  frequency which has **space complexity** of  $O((k + F_2/m_k^2) \log n)$ 
  - Used for the streaming model

## Model

Let's make some ground rules



9

Distributed Computation of The Mode

10 May 2010

## Model

- Graph  $G = (V, E)$ :  $V$  – node set,  $E$  – edge set,  $D$  – graph diameter
- The total number of nodes is denoted  $n$
- Each node  $v$  in  $V$ , stores one / more elements:  $e_i$
- There are  $K$  possible elements
- The elements are ordered
- The frequency of each element  $e_i$  throughout the graph is denoted  $m_i$

10

Distributed Computation of The Mode

10 May 2010

## Model – Cont.

- The upper-bound for the frequencies is denoted  $M$
- The total number of elements is denoted  $m$
- The total number of distinct elements is denoted  $k$
- The algorithms are **synchronous**
  - Time complexity – number of rounds needed until every node terminates

11

Distributed Computation of The Mode

10 May 2010

## Model – Cont.

- We define “frequency moments”:

DEFINITION 3.1. (Frequency Moments) The  $\ell^{\text{th}}$  frequency moment  $F_\ell$  of a multiset containing  $m_i$  elements of type  $e_i \in [K]$  is defined as  $F_\ell = \sum_{i=1}^k m_i^\ell$ .

- Notes:

- $F_0 = k, F_1 = m$

12

Distributed Computation of The Mode

10 May 2010

## Model - Communication

- The computation is done with the exchange of messages from nodes
- Nodes can communicate if and only if they're connected in the graph
- The size of each message is restricted to  $B \in O(\log K + \log M + \log n)$

## Model - BFS Pre-computation

- We assume that a BFS spanning tree rooted at the node initiating the algorithm was pre-computed
- Since the time-complexity for BFS is  $2D$ , and all our bounds will be at least linear in  $D$  – this is not critical

## Solution(s)

Let's get down to business



## Solution

- Two algorithms are shown:
  - A deterministic algorithm
  - A randomized algorithm
- We'll describe and analyze the algorithms

## Deterministic Algorithm

Offers certainty – that costs!



17

Distributed Computation of The Mode

10 May 2010

## Deterministic Algorithm

- Starts at “leaves” of spanning-tree
- Each “leaf” sends element-frequency pairs  $(e_i, m_i)$  to its parent, for each of its elements
- This is done in an increasing order on the elements

18

Distributed Computation of The Mode

10 May 2010

## Deterministic Algorithm

- When an elements received the “i pair” (or higher) from all its leaves it sums up the pairs and forwards it to its parent
- The time complexity is intuitively  $O(D + k)$ 
  - Forwarding of the “i pair”s is done in parallel
  - Keep this mind ;-)

19

Distributed Computation of The Mode

10 May 2010

## Deterministic Algorithm – Lower Bound

- We first look at the “set disjointness” problem:
  - Input: Two sets  $S_1$  and  $S_2$  for which,  $|S_1| = |S_2| = L$ , and  $|S_1 \cap S_2| \in \{0, 1\}$
  - Output: Whether the two sets are disjoint or not

20

Distributed Computation of The Mode

10 May 2010

## Deterministic Algorithm – Lower Bound – Cont.

- A **reduction** from this problem to ours can be done by simply addressing the two sets as two nodes connected, creating a graph



- The sets are disjoint **if and only if** the mode's frequency is 2

21

Distributed Computation of The Mode

10 May 2010

## Deterministic Algorithm – Lower Bound Cont.

- The lower bound for the amount of bits exchanges for the problem is  $\Omega(k)$  (with sets holding elements of cardinality  $k/2$ )
- Each message can only be of  $B$  bits; This implies a time-complexity lower bound of  $\Omega(k / B)$
- Since  $D$  is also a natural bound for this time-complexity, the time lower bound for this problem is proven to be  $\Omega(D+k/B)$

22

Distributed Computation of The Mode

10 May 2010

## Randomized Algorithm

Monte Carlo never seemed so attractive



23

Distributed Computation of The Mode

10 May 2010

## Randomized Algorithm

- Each node has **the same** hash function which maps each element to one of two bins -1 or 1:  $h_i : A \rightarrow \{-1, 1\}$
- Every round:
  - A node calculates how many of its elements were mapped to each bin in counter:  $c_0$  (for -1 bin) and  $c_1$
  - It waits for a corresponding tuple  $(c_0, c_1)$  from all its children
  - Once all tuples arrived, they're accumulated with the node's into a message which is sent to the node's parent

24

Distributed Computation of The Mode

10 May 2010

## Randomized Algorithm – Cont.

**Algorithm 1** *countElementsInBins*( $\mathfrak{h}$ ) at node  $v$  with multiset  $\langle e_1, \dots, e_t \rangle$ :

```
1:  $c_0 = |\{e_i : \mathfrak{h}(e_i) = -1\}|$ 
2:  $c_1 = |\{e_i : \mathfrak{h}(e_i) = +1\}|$ 
3: if  $\Gamma(v) = \emptyset$  then
4:   send  $\langle c_0, c_1 \rangle$  to  $p(v)$ 
5: else
6:   for all  $v_j \in \Gamma(v)$  in parallel do
7:      $\langle c_0^{(j)}, c_1^{(j)} \rangle = \text{countElementsInBins}(\mathfrak{h})$ 
8:   send  $\langle c_0, c_1 \rangle + \sum_{j \in \Gamma(v)} \langle c_0^{(j)}, c_1^{(j)} \rangle$  to  $p(v)$ 
```

## Randomized Algorithm – Cont.

- Once the tuple  $(c_1, c_0)$  has been calculated it's distributed through out the graph
- Each node has a counter  $c(e_i)$  for each of the elements it's holding
- Each node which gets the tuple increases all counters of elements mapped to the largest bin by  $|c_1 - c_0|$

## Randomized Algorithm – Cont.

- This would be done  $r_1$  times with  $r_1$  **different** hash functions chosen randomly
  - After this first phase, the number of “mode candidates” would be reduced significantly
- The largest  $r_2$  elements are “gathered” and the mode is selected from them
  - The frequency for each of these candidates would be computed and the mode would be found

## Randomized Algorithm – Cont.

**Algorithm 2**  $\mathcal{ALG}_{mode}$

```
1: mode =  $\emptyset$ , freq =  $-\infty$ 
2: Phase (1):
3: for  $i = 1, \dots, r_1$  in parallel do
4:    $\langle c_0, c_1 \rangle = \text{countElementsInBins}(\mathfrak{h}_i)$ 
5:   distribute( $\langle \langle c_0, c_1 \rangle, \mathfrak{h}_i \rangle$ )
6: Phase (2):
7:  $\langle e_1, \dots, e_{r_2} \rangle = \text{getPotentialModes}(r_2)$ 
8: for  $i = 1, \dots, r_2$  in parallel do
9:    $m_i = \text{getFrequency}(e_i)$ 
10:  if  $m_i > \text{freq}$  then
11:    mode =  $e_i$ , freq =  $m_i$ 
12: return mode
```

## Randomized Algorithm – Cont.

- Regarding the fact that the  $r_1$  calculations on the  $r_1$  hash of functions is done in parallel, it's upper bound is  $O(D + r_1)$ 
  - Similar to what we've seen in the deterministic algorithm time-complexity upper bound
- On the same principle – the “flooding” of the top  $r_2$  candidates takes  $O(D + r_2)$

29

Distributed Computation of The Mode

10 May 2010

## Calculating $r_1$ and $r_2$

- We still have to find proper values for  $r_1$  and  $r_2$  to optimize the time-complexity on our algorithm
- We first calculate the expressions for these, then finalize by estimating inner expressions

30

Distributed Computation of The Mode

10 May 2010

## Calculating $r_1$ and $r_2$ – Cont.

LEMMA 4.2. If  $r_1 \in O(F_2/m_1^2 \log(k/\epsilon))$  then  $\forall e_i : m_i < m_1/2$  it holds that  $c(e_i) < c(e_1)$  with probability at least  $1 - \epsilon$ .

THEOREM 4.3. The time complexity of  $\mathcal{ALG}_{mode}$  to compute the mode with probability at least  $1 - \epsilon$  on an arbitrary graph  $G$  of diameter  $D$  is

$$O\left(D + \frac{F_2}{m_1^2} \log \frac{k}{\epsilon}\right).$$

31

Distributed Computation of The Mode

10 May 2010

## Calculating $r_1$ and $r_2$ – Cont.

- We'll define –  $\check{Y} = \{e_i : m_i \geq m_1/2\}$
- It's clear that  $|\check{Y}|(m_1/2)^2 \leq \sum_{e \in \check{Y}} m_i^2 \leq F_2 \rightarrow |\check{Y}| \leq 4F_2/m_1^2$
- And according to Lemma 4.2 if  $r_1 = 32F_2/m_1^2 \ln(2k/\epsilon)$  then  $\check{Y}$  has the mode with probability of  $1 - \epsilon$
- So – choosing  $r_2 = 4F_2/m_1^2$ , and the above  $r_1$  would have **both phases** done after  $O(D + F_2/m_1^2 \log(k/\epsilon))$  with the mode found with probability  $1 - \epsilon$

32

Distributed Computation of The Mode

10 May 2010

## Sizes Estimations

- After finding the required sizes for  $r_1$  and  $r_2$ , we can see that an estimation for  $k$ ,  $m_1$  and  $F_2$  is still needed

## Estimation of $F_2$

- Algorithm (Alon et al.)
- Estimates  $F_2'$  which deviates from  $F_2$  by at most  $\lambda F_2$  with probability  $1 - \epsilon$
- Calculates the values  $X_j := \sum_{i=1}^k h_j(e_i) m_i$  which we can compute using our counters as  $X_j = c_1^j - c_0^j$
- Shows that using  $s := \frac{32 \log(\frac{1}{\epsilon'})}{\lambda^2}$  “four-wise independent” hash-functions is suffice

## Estimation of $F_2$ - Cont.

- As we're using  $\frac{32 \log(\frac{1}{\epsilon'})}{\lambda^2}$  hash functions, similar to before (the “me'asef” principle) – the computation would take  $O(D + \log(1/\epsilon))$

## Estimating $k$

- Using mapping of elements to bit strings
- Longest 0's prefix is used to estimate  $\log_2 k$
- Several runs are done to bound variance
- Distributed version has the longest prefix of 0 bits for each *hash function* accumulated and used to compute an estimate  $k'$  in  $O(D)$  time

## Estimating $m_1$

- Note that we have estimators for  $k$  and  $F_2$
- Finding an estimator for  $m_1$  is supposedly the most “problematic”
- Recall that we set  $r_1 = 32F_2/m_1^2 \ln(2k/\epsilon)$
- We use a dynamic system to estimate  $m_1'$

37

Distributed Computation of The Mode

10 May 2010

## Estimating $m_1$ - Cont.

- After each distribution of a tuple, we determine the frequency of the most-“frequent” element, in  $O(D)$  time
- If it’s larger than its’ previous – it’s the new  $m_1'$
- We keep doing this while  $T > 32F_2'/m_1'^2 \ln(2k'/\epsilon)$  where  $T$  denotes the number of rounds made

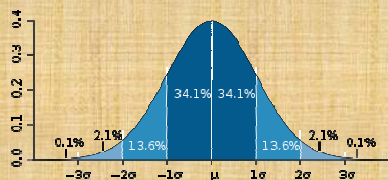
38

Distributed Computation of The Mode

10 May 2010

## Further Studying of Freq. Dist. Consideration

Is it really that good?



39

Distributed Computation of The Mode

10 May 2010

## Reduced Lower Bound

- To further show how certain freq. distributions might improve the runtime dramatically, a construction is made to show the following statement
- For every natural  $m_1, F_5$ , a graph  $G$  with diameter  $D$  can be created, for which finding the mode requires  $\Omega(D + F_5/(m_1^5 B))$  rounds
- This is also done using a reduction from the *set disjointness problem*

40

Distributed Computation of The Mode

10 May 2010

## Power Law Distribution

- A widely studied distribution
  - $p(x) \propto x^{-\alpha}$  for some constant  $\alpha > 0$
- We'll study a specific case:  $m_i = \frac{1}{i^\alpha}$
- Let  $m = \sum_{i=1}^k m_i$ ; It holds that:
  - For  $\alpha < 1, m \in \Theta(k^{1-\alpha})$
  - For  $\alpha = 1, m \in \Theta(\log k)$
  - For  $\alpha > 1, m \in \Theta(1)$

41

Distributed Computation of The Mode

10 May 2010

## Power Law Distribution – Cont.

- We can obtain the following upper bound(s) on the running of the randomized algorithm:

$$T \in \begin{cases} O(D + k^{1-2\alpha} \cdot (\log k + \log(1/\epsilon))), & \text{if } \alpha < 1/2 \\ O(D + \log k \cdot (\log k + \log(1/\epsilon))) & \text{if } \alpha = 1/2 \\ O(D + \log k + \log(1/\epsilon)), & \text{if } \alpha > 1/2. \end{cases}$$

42

Distributed Computation of The Mode

10 May 2010

## Conclusion

What do we really take from all of this?



43

Distributed Computation of The Mode

10 May 2010

## Conclusion

- We've seen the mode can be computed **deterministically** in  $O(D+k)$  time – with no regard to frequency distribution
- A randomized algorithm was presented to calculate the mode with high probability in time of  $O(D + F_2 \log k / m_1^2)$

44

Distributed Computation of The Mode

10 May 2010

## Conclusion – Cont.

- A possible lower bound:  $\Omega(D + F_5/(m_1^5 B))$  is shown, which significantly emphasizes the potential of considering the frequency dist
- We also consider specific power law distributions which reduce the running time to a **poly-logarithmic** expression